29 | 网络和邮件库：定时收发邮件，减少手动操作

2021-04-17 尹会生

Python自动化办公实战课 进入课程 >

讲述：尹会生
时长 21:14 大小 19.45M

你好，我是尹会生。

相信在你的日常办公工作当中，对邮件肯定早就不陌生了。我们通过邮件既可以发送和接收正式的公文，也能够利用邮件编写周报、月报，以及订阅定期发布的新闻或者期刊，等等。

在这么多的应用场景当中，你会发现有些**收发邮件的工作是周期性**的。那我们就可以利用Python，将这些重复的邮件收发工作进行自动化。

设想一下：在我们接收邮件的时候，我们可以定时监察邮箱中的邮件，例如根据邮件中特定的主题，来自动判断是否为重要邮件。如果是的话，可以通过 Python 调用钉钉等即时通讯软件马上通知自己，实现邮件的额外通知功能。

还有在发送邮件的场景中，如果你发现周报、月报等邮件模版是可以通用的，那你可以利用 Python 的字符串编写来实现邮件内容的自动替换功能，让你在使用邮件发送周报、月报工作中节约大量的时间。

那么今天我就教你如何使用 Python 的 yagmail、imaplib 两个库，分别实现邮件的自动发送和自动接收功能，并利用正则表达式、字符串和变量功能，来替代手工的重复工作。

自动收邮件

我们先从如何自动收邮件开始学习。今天的案例是这样的：我希望能每隔五分钟检查一次收件箱，判断收件箱中是否有 30 天内未读的邮件。并利用正则表达式根据邮件主题判断其中是否包含“故障”这一关键字。如果包含的话，就通过钉钉等即时通讯工具通知到我，实现高优先级邮件处理的功能。

要想利用 Python 的正则表达式判断邮件主题是否出现了“故障”关键字，你必须要让 Python 实现**邮件接收和主题读取**功能。在 Python 中，**poplib 和 imaplib 库都**支持邮件的接收协议，可以让我们登陆服务器接收邮件，从而实现邮件接收和主题读取。那这两个库该选择哪一个呢？

imaplib 库支持 IMAP 协议，而 poplib 库支持 POP3 协议，IMAP 协议在支持双向操作的功能上更加强大，并且能把客户端对邮件的删除等操作同步到服务端，也能把服务端对邮件删除的操作同步到客户端。与 POP3 协议只能把服务端的操作单向同步给客户端相比，会更加灵活。所以我在本讲中，就以 imaplib 库为例，为你讲解通过 IMAP 协议进行邮件的自动接收。

我们在确定采用 IMAP 协议接收邮件之后，接下来就要按照 IMAP 协议的要求，编写一个从邮件服务器下载邮件并分析邮件主题的代码。获取邮件主题的代码分为三个主要步骤，分别是**指定邮件服务器的 IMAP 地址和端口、验证用户名和密码的正确性以及下载邮件到本地并解析邮件得到邮件主题**。我们依次来学习一下。

获取邮件主题

第一步是**指定邮件服务器的 IMAP 地址和端口**。大部分对邮件安全比较重视的公司，为了防止黑客暴力发现邮件服务器用户的弱口令密码，默认是将 IMAP 服务的功能关闭的。你

需要联系邮件服务器管理员或通过网页管理功能打开 IMAP 服务，允许你在家里连接 IMAP 服务器。

我以 QQ 邮箱为例，打开 IMAP 服务的方法是在QQ 邮箱的网页端登陆成功后，通过设置 - 账号 -IMAP 服务，打开 IMAP/SMTP 服务。打开功能后，可以参考官方文档将 IMAP 服务器的地址指定为：“imap.qq.com”，“使用 SSL”保证数据传输过程的安全，并将连接 IMAP 服务器的端口指定为“993”。打开 IMAP 服务的截图和官方文档的截图如下。

第二步是**使用用户名密码登录**。当服务器允许你从远程使用 IMAP 协议登陆服务器接收邮件后，就可以使用 Python 的 imaplib 库进行连接和登陆了。

imaplib 库是 Python 的内置库，连接服务器可以使用 IMAP4_SSL() 函数，登录可以使用 login() 函数，连接和登陆的代码如下：

复制代码

```
1 import imaplib
2 conn = imaplib.IMAP4_SSL(host="imap.qq.com", port = 993)
3 conn.login("username@qq.com","password")
4 print(conn.list())
```

在这段代码中的第三行，你需要把“username 和 password”替换为你的用户名和密码，替换之后才能正常登陆。如果登陆成功，可以通过 list() 函数查看邮箱中默认包含了哪些文件夹，默认的邮件都被放在“INBOX”文件夹中，而“INBOX”就是我们经常使用的收件箱。

如果没有登陆成功，在运行代码后会被提示连接超时或密码错误，这个时候你就需要根据错误提示，进一步优化你的网络质量或使用正确的密码。

最后一步是**解析邮件主题**。当你成功登录邮件服务器之后，你并不能直接读取邮件的内容，必须要将邮件下载到本地才能对邮件内容进行解码和内容查看。这时，你如果对收件箱中的邮件进行查询，只能得到未读邮件的唯一 ID，我们称它为“邮件 ID”，你需要**通过**

IMAP 的 fetch() 命令将邮件 ID 对应的邮件内容下载本地后，才能进行解码，解码后才能真正取得邮件的主题、内容和附件等邮件里的具体内容。

你还要注意的是，通过邮件 ID 下载的邮件内容需要**解码两次，才能看到邮件的主题**。这是因为每一封邮件都采用了邮件的标准编码方式 MIME 编码，MIME 编码可以让邮件在服务器和客户端直接实现正常的传输，但是你无法查看使用了 MIME 编码之后的邮件内容，因此需要先对邮件的 MIME 编码进行解码。

而第二次解码呢，是**把 MIME 编码的邮件解码后的内容，转换成符合你当前操作系统的编码**，否则在查看邮件主题时会出现乱码，无法使用正则表达式进行内容匹配。

总结来说，在 Windows 中，默认的编码为“GBK”编码，mac 为“UTF-8”编码。你需要把内容按照 Python 所在的操作系统再解码一次，这样才能得到正确的邮件标题，之后才能使用正则表达式处理我们得到的邮件主题。

使用的函数

虽然获得邮件的主题的步骤比较繁琐，无法一次性得到邮件主题，不过你不用担心，因为**每个处理步骤中只需要使用一个函数就可以搞定**了。那么接下来我先把从进入收件箱到取得主题的完整执行过程的代码提供给你，然后再为你具体讲解每个函数的作用。

复制代码

```
import email

# 默认为INBOX
conn.select("INBOX")
# 搜索邮件，ALL为全部，可以按照发件人使用FROM过滤，也可以使用日期过滤
_, data = conn.search(None, 'unseen')

for mailid in data[0].decode().split(" "):
    # 取回每一封未读邮件的内容
    # data = [b'1 2 3 4 5']
    _, maildata = conn.fetch(str(mailid), '(RFC822)')
    # 对每一封邮件的内容进行解析
    msg = email.message_from_string(maildata[0][1].decode('utf-8'))
    # 取得标题
    subject_tmp = msg.get('subject')
    # 为标题解码
    sj_decode = email.header.decode_header(subject_tmp)[0][0]
    #打印每一封标题
    subject = sj_decode.decode('utf-8')

```

```
21      print(subject)
22
23      # 将邮件标记为已读
        conn.store(mailid, '+FLAGS', '\\Seen')
```

我来为你依次解释一下 imaplib 是如何读取邮件并得到邮件主题的。

首先，我们需要**从收件箱中找到 30 天内未读的邮件 ID**，通过邮件的 ID 才能从 IMAP 服务器下载邮件的内容。

我在代码第 4 行，使用了 select() 函数，指定要读取的文件夹为收件箱“INBOX”；再利用第 6 行的 search() 函数的“unseen”参数，来取得 30 天内未读的邮件。这里的“INBOX 和 unseen”都是 IMAP 协议定义的关键字，Python 会将它们转译为 IMAP 的语法，并发送给服务器，而服务器则会把 30 天以内未读邮件的 ID 以列表形式返回，并把 ID 以字节方式存放在 data 列表的第一个元素中，data 列表的具体格式，你可以参考第 10 行注释。

接下来，我们需要**根据每个邮件 ID 得到邮件的主题**。由于邮件 ID 为字节类型，因此我将它转换为列表之后，使用 for 循环进行遍历，把每个邮件 ID 用 mailid 变量进行了保存。同时，我在第 11 行通过 fetch() 函数，使用邮件 ID 向服务器发起请求，得到该邮件的所有数据。

最后，我们**把从服务器得到的邮件内容进行解析、取出标题部分，并进行解码**。当你使用 fetch() 函数取得了邮件的内容后，如果使用 print() 进行输出，会发现你无法看到邮件里真正的内容。为什么会出现这种情况呢？

原因就在于邮件采用的是 MIME 类型，这种类型是邮件的标准格式，需要专门的工具进行内容的解析。就像你无法通过记事本查看一张图片一样，通过记事本只能看到图片中混乱的数据，无法得知图片上面的颜色和内容。因此我需要再使用一个标准库 email 的 message_from_string() 方法，对 MIME 类型进行解析。

解析之后，你就能够得到主题、内容、附件等邮件的不同部分了，由于我在当前案例需要提取邮件的主题，所以使用第 15 行的 get() 函数通过参数“subject”取得了当前邮件的头部信息，并利用 decode_header() 取得了邮件的主题。

为了能够在 Mac 系统上也可以进行处理，我将主题采用“utf-8”编码进行解码后，就能够正常显示汉字了。如果你想要判断该主题是否包含“故障”关键字，可以使用我们多次使用到的正则表达式，使用 re.search('故障', subject) 进行正则匹配，并且你还可以增加钉钉通知、短信通知、自动修复故障等等各种自动化操作。

在学习了自动接收邮件并对主题进行判断的功能之后，还有两点需要你注意，这也是初次使用 imaplib 库的同学最容易犯的两个错。

第一个是**如果你没有将存储在 IMAP 服务器上的邮件标记为已读，会导致自动接收邮件程序重复处理该邮件**。由于 fetch() 函数的功能是从服务器下载邮件内容，并对邮件进行自动化处理，所以服务器上的邮件状态仍然为“未读”状态。这会导致你的程序陷入死循环，对匹配的主题进行无限重复的处理。

为了避免这一问题，你应该在处理完当前的邮件后，使用“conn.store(mailid, '+FLAGS','\seen')”方法，将当前操作的 mailid 在服务器设置为已读邮件。这样每个邮件就会只被处理一次了。

另一个经常出现的问题是，当你的所有邮件都为已读状态时，应当在遍历邮件 ID 功能前增加对 data 变量的判断，避免向服务器提交空 ID，导致运行到 fetch() 函数时，服务器接收空 ID 报错。

具体的操作是：你可以在得到 data 变量后，使用 if 判断该变量是否为“None”，如果为 None，则本次执行到此结束。如果有未读邮件，则再将邮件 ID 通过 fetch() 提交到服务器进行处理。

当你已经掌握了自动接收邮件的步骤之后，再来学习自动发送邮件就非常简单了。自动发送邮件采用了 SMTP 协议，而且也需要指定服务器地址、用户名、密码以及收件人、主题、内容和附件。由于发邮件和接收邮件的大部分概念相同，所以我们可以对比接收邮件来学习实现自动发送邮件的步骤。

自动发邮件

和自动接收邮件类似，自动发送邮件的步骤也是三个，分别是连接邮件服务器、编写邮件正文和发送邮件。

连接邮件服务器

在邮件服务的协议规范中，规定发送邮件采用的是 SMTP 协议，因此，在自动发送邮件这一步，我们需要采用和 imaplib 不同的包实现。

在标准库中发送邮件的包叫做 smtplib，由于 smtplib 需要配置较多的通用参数，所以还有一个对它进行了更高级的封装的第三方库 yagmail 库。yagmail 库将大部分的默认参数在底层实现了，发送邮件时，你只需要关注必须填写的服务器 IP、用户验证以及邮件的内容即可。

yagmail 第三方库的安装包和它同名，那么你可以使用 pip 命令直接安装，安装成功后把它导入并连接服务器即可。连接 SMTP 服务器的代码如下：

复制代码

```
import yagmail
conn = yagmail.SMTP(
        user="username@qq.com",
        password="password",
        host="smtp.qq.com",
        port=465
        )

```

yagmail 库使用 SMTP() 函数与服务器建立连接，并在连接时指定用户名、密码、主机地址以及端口。

这里需要注意的是，SMTP() 函数通过默认参数“smtp_ssl=True”使用了 SSL 协议，如果你所使用的邮件服务器采用了不同版本的 SSL 传输加密协议，你需要先将默认端口从 465 改为 587。如果没有提示连接异常，表明建立连接是成功的，接下来就可以为这封邮件编写内容了。

编写邮件正文

编写邮件正文时，可以采用我们学习过的 f-string**字符串的形式**来存放邮件的内容。例如你经常要发送的周报、月报都是相同的邮件格式，不同的数据内容或文字。这时候可以使用 f-string 字符串的变量替换功能，将格式编写为 f-string 的字符串，再将每次变动的内容使用变量进行替换，它的代码格式如下：

复制代码

```
content = "内容填充"
body = f"模版 {content}"
```

当你编写好邮件的正文后，需要使用 send() 函数来发送邮件。**send() 函数一般会使用四个参数，按照参数定义的顺序，它们分别是收件人邮箱、主题、邮件正文和附件。**我将这四个部分依次作为 send() 函数的参数后，就可以将邮件发送到 SMTP 服务器了。

这里我有一个小的建议，我会建议你先把邮件发给自己，如果出现发送失败，或发送内容与自己期望不符时，更方便对内容进行调整。我把发送命令和发送成功后的截图贴在下面，供你参考。

复制代码

```
conn.send("receiver@qq.com", "主题", body, "one.jpg")
```

conn.send("receiver@qq.com", "主题", body, "one.jpg")

这就是利用 yagmail 实现自动发送邮件的完整过程，掌握之后，你可以把上节课学习的定时任务，以及自动生成图形的 matplotlib 库结合起来使用，将发送邮件功能定义为一个函数，从而实现周报和月报的自动发送功能。

小结

最后，我来为你总结一下这一讲的主要内容。在本讲，我使用了 yagmail 库、imaplib 库以及 email 库实现了邮件自动收发的功能。与 Foxmail 和 Outlook 比起来，使用 Python 实现的邮件客户端，能够在收取邮件后对主题等元素自行判断，并与正则表达式、IM 通知等其他工具组合，实现更加自动化的功能。

利用 yagmail 自动发邮件前，你还可以为你的周期发送的邮件指定模版，通过 Celery 实现定时发送和周期发送邮件。

除了可以自动收发邮件外，我还为你详细讲解了接收邮件的处理过程，这一过程遵循了 IMAP 的协议规范，决定了代码编写的先后顺序，**如果邮件接收或发送是你自动化工作中主要优化的工具，那么我建议你用更多的时间来掌握 IMAP 与 SMTP 协议规范。**

当你掌握了 IMAP 协议能够支持哪些操作以及不能支持哪些操作之后，才能更好地进行邮件 API 的学习。对于 IMAP 支持的功能，你可以参考官方文档，找到函数及其参数，对于没有支持的功能，你需要自己实现自定义的函数。

最后的最后，除了微信、钉钉外，邮件可以说是我们职场中使用最广泛的通讯工具了。并且也是我们工作中最正式的通讯工具。因此我建议你能够多练习怎么通过 Python 更加熟练地自动化收发邮件，相信我，这会为你的工作带来更高效的输出。

思考题

按照惯例，我来为你留一道思考题，如果我希望每周六 10 点整，能够自动的将 C 盘上的一个目录作为邮件的附件发送到一个指定邮箱，你会使用哪些库来实现，你能否将实现的思路用自己的语言讲出来呢？

欢迎把你的想法和思考分享在留言区，我们一起交流讨论。也欢迎你把课程分享给你的同事、朋友，我们一起做职场中的效率人。我们下节课再见！

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 28 | Celery库：让计算机定时执行任务，解放人力

下一篇 30 | 怎么快速把任意文件格式转成PDF，并批量加水印?

精选留言 (1)

写留言

Soul of the Drago...

2021-04-17

老师，请问我在运行“conn.select("INBOX")”这一句的时候，为什么会出现“error: command SELECT illegal in state NONAUTH, only allowed in states AUTH, SELECTED” 这样的报错呢?

展开

作者回复: 你好，这就是我在文章中提过的，你需要先验证通过，才能选择“服务器上的文件夹”。 请检查你的用户名密码是否正确